MESO*CENTRE
Aix*Marseille Université

UTILISATION DES RESSOURCES DU Mésocentre

Annie Clément
Matvey Sapunov

01/12/2014

Aix*Marseille université

Aix*Marseille université

Programme :

- **Présentation du Mésocentre**
- **Comment se connecter aux machines ?**
- **L'environnement de l'utilisateur**
- **Les logiciels**
- **Le gestionnaire de ressources : OAR**
- **Questions / Réponses**

MESO*CENTRE
Aix*MarseilleUniversité

Présentation du mésocentre : les services

Créé en 2012, il offre les services :

- ✓ **Accès**
- ✓ **Calcul**
- ✓ **Visualisation**
- ✓ **Assistance**
- ✓ **Animation scientifique**
- ✓ **Exploitation, Administration, et veille prospective**

Présentation du mésocentre : les équipements

- **Total de 1384 cœurs :**
  - 1152 cœurs de calculs (96 noeuds x 12)
  - 188 cœurs à forte mémoire (SMP)
  - 16 nœuds GPU
  - 16 nœuds sur carte Xeon PHI
  - 12 nœuds de visualisation
- **Espace de stockage GPFS (300 téraoctets)**
- **Réseau d'interconnexion des équipements Infiniband**

Présentation du Mésocentre : les projets

Les Types :

- **A : Projet à durée limitée de découverte et/ou portage, environ 5 000h**
- **B : Allocation longue durée, entre 10 000 et 400 000 heures, 3 sessions annuelles d'examen des demandes**
- **mesochallenge : Réservation ponctuelle de la majorité des ressources sur un temps très court**

Les demandes d'allocation de ressources se font en ligne à partir du site web du mésocentre, rubrique déposer un projet

Présentation du Mésocentre : règles d'utilisation

- **Charte : respect de la législation et de l'éthique**
- **Communication des publications et remerciements**
- **Stockage de données et installation de logiciel sous la responsabilité des usagers**
- **Les comptes d'accès sont nominatifs, personnels et non cessibles.**

Comment se connecter au Mésocentre ?

- **Faire partie d'au moins un projet et avoir un compte utilisateur**
  - identifiant et mot de passe
  - quota d'heures par projet

- **Se connecter à la frontale en accès ssh :**
  - Linux / macOS – depuis un terminal :

    ssh identifiant@login.ccamu.u-3mrs.fr
  - Windows : logiciel type putty

- **Mot de passe et clé ssh**

Environnement utilisateur

- **Espaces de stockage alloués :**
  - /home/utilisateur : stockage persistant, NFS, 5Go, avertissement
  - /tmp : stockage temporaire, local au noeud
  - /scratch : calcul, GPFS, 10To, avertissement

Ces espaces sont sous la responsabilité des usagés, il n'y a pas de sauvegardes automatiques => *Sauvegarde usagers sur une machine locale par scp, sftp, rsync*

- **Utilisation des modules :** permet de configurer, à la demande, l'environnement de l'utilisateur. Par exemple définir le compilateur à utiliser.
  - *module avail* liste des modules disponibles
  - *module list* liste les modules chargés dans l'environnement
  - *module load intel/15.0.0* charge le compilateur intel 15.0
  - *module unload intel/15.0.0* décharge le compilateur intel 15.0
  - *module purge* décharge tous les modules de l'environnement

Environnement utilisateur

- **Le karma :** « note » propre à chaque utilisateur qui fluctue en fonction des calculs réalisés et dont l'ordonnanceur des travaux tient compte

- Les quotas : heures consommées par projet / espace disque

***----------**

| On project 14bXXX : 9122.9 hours have been consumed

| You are using 1072/4882 MB (21%) on /home

| You are using 0.00/9.00 TB (0%) on /scratch

***----------**

Les bibliothèques, logiciels, utilitaires

- **Mis à disposition :**
  - Installés sur partition /softs
  - Modules associés

- **Installés par les utilisateurs :**
  - Sur l'un de leurs espaces disque
  - Sous leur responsabilité

Si le logiciel souhaité n'est pas proposé, le mésocentre peut étudier son acquisition et son installation

Le gestionnaire de ressources OAR

- **OAR** is a framework written in Perl which glue together different tools
- Batch Sceduler which build on top of high level tools
  - Relational database (**MySQL** or **PostgreSQL**)
  - Confinement system mechanism **cpuset**
  - Scalable exploiting tool **Taktuk**
- A **cpuset** is a kernel object to which can be assigned a set of CPUs and memory nodes, and this object can be attached to processes
  - Hardware resource management
    - Allocation
    - Confinement
    - Cleaning
- **TakTuk** is a tool for deploying parallel remote executions of commands to a set of remote nodes
  - Check nodes states
  - Commands execution on the nodes

OAR features

- Batch and Interactive jobs
- Multi-queues with priority
- Reservation
- Support of moldable tasks
- Epilogue/Prologue scripts
- Suspend/resume jobs
- Checkpoint/resubmit
- Hierarchical resource requests (handle heterogeneous clusters)
- Full or partial time-sharing.
- Licenses servers management support.
- Best effort jobs : if another job wants the same resources then it is deleted automatically

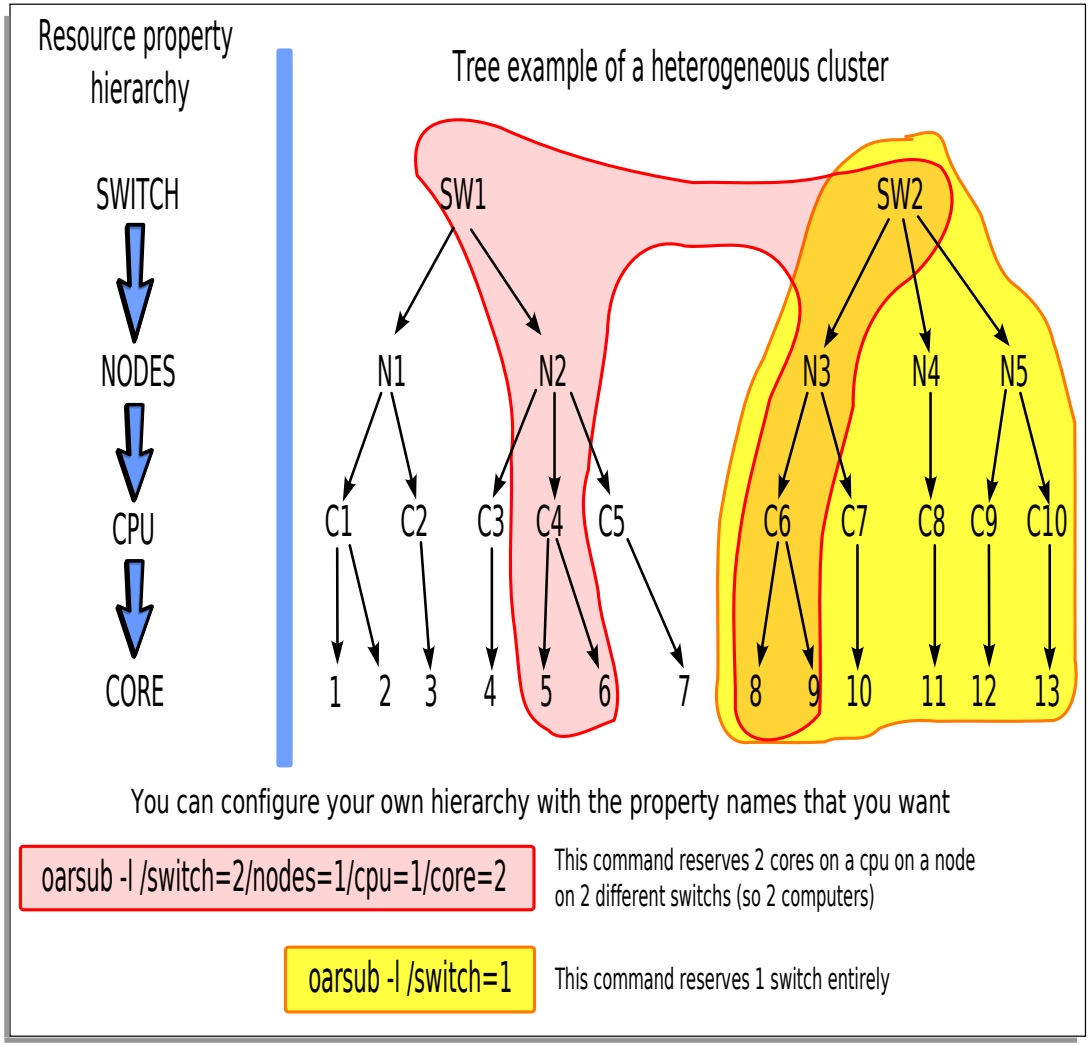
OAR architecture

- **server** node which runs the oar server daemon and a database which store all job related information
  - Key component
- **frontend** node on which you will be allowed to login and to reserve computing resources
  - login.ccamu.u-3mrs.fr
- **computing** nodes on which the jobs will run
  - node001 – node096
  - smp001 – smp004
  - visu
  - phi001
  - gpu001
- **visualization** node on which all the visualization web interfaces are accessible
  - Not available from external network

OAR server modules

- **Almighty**: The OAR server. Overmind
  - Listen a TCP/IP socket for information or a command from OAR user program or a module
  - Command execution thread. Launch modules in correct order
- **Sarko**: sanity module
  - Check walltime of the running jobs and ask to frag them if they had expired
  - Detect if fragged jobs are really fragged otherwise asks to exterminate them
  - Can change "Suspected" resources into "Dead"
- **Judas**: Notification and logging module
  - Notification mechanism based on e-mails and scripts
  - Redirects STDOUT and STDERR into a log file. Set log levels for the messages
- **Leon**: Job deletion. Modes :
  - normal: Leon tries to connect to the first node allocated for the job and terminates the job
  - exterminate : clean up the database entry for the job
- **Runner**: runs an OAR job, set jobs states and treat job errors and reservations
- **NodeChangeState**: module is in charge of changing resource states and checking if there are any jobs running on the resources
- **Scheduler**: module checks for each reservation jobs if it is valid and launches the job at the right time
- **Hulot**: module is responsible of the advanced management of the standby nodes

Resource allocation



Wanted resources have to be described in a hierarchical manner

Complete syntax :
"{ sql1 }/prop1=1/prop2=3+
{sql2}/prop3=2/prop4=1/prop5=1
+...,walltime=HH:mm:ss"

walltime is always the last parameter

Examples :

- nodes=1/core=4,walltime=80:00:00
- core=2,walltime=168:00:00
- nodes=2,walltime=30:00:00
- host=16,walltime=47:59:00
- nodes=5/core=6,walltime=1:59:00

Fine resource allocation

Fine resource selection is done by using properties attributed to a resource

- SQL syntax
  - "cluster = 'YES' AND shortnode = 'NO' AND host NOT IN 'gpu001'"
  - "((smp='YES' and host='smp004') AND shortnode = 'NO') AND host NOT IN ('gpu001')"
  - "smp and nodetype='SMP512Gb'"

- Shortcuts
  - cluster → "cluster = 'YES'"
  - smp → "smp = 'YES'"
  - visu → "visu = 'YES'"
  - gpu → "gpu = 'YES' AND visu = 'NO'"
  - phi → "phi = 'YES'"

OAR resource states

- oarnodes – command to display resource related information
- OAR resource states : oarnodes -s

```
node002
    13 : Alive
    14 : Alive
    15 : Alive
    16 : Alive
    ...
    23 : Alive
    24 : Alive
```

- **Alive**: the resource is ready to accept a job.
- **Absent**: the oar administrator has decided to pull out the resource. This resource can come back.
- **Suspected**: OAR system has detected a problem on this resource and so has suspected it. This resource can come back automatically or manually.
- **Dead**: The oar administrator considers that the resource will not come back and will be removed from the pool

Resource properties

Display available ressource properties : oarnodes -r **resource_id**

- freq=3.07
- cpuset=9
- model=X5675
- smp=NO
- phi=NO
- gpudevice=0
- cpu=180
- swib=9
- gpu=NO
- ib=YES
- board=90
- mem=24
- type=default
- shortnode=NO
- gpunum=0
- deploy=NO
- core=1080
- cluster=YES
- ip=192.168.71.90
- visu=NO
- available_upto=2147483646
- nbcores=12
- nodetype=Westmere
- desktop_computing=NO
- last_available_upto=2147483646
- network_address=node090
- host=node090
- vgldisplay=:0.0
- last_job_datebesteffort=YES
- vncdisplay=0

Project in OAR

A user can participate in different scientific activities. To simplify accounting of the consumed resources by activity a notion of the project has been introduced since OAR version 2.5

Each user in mesocentre has a corresponding project

- One user can be registered in several projects

Attributing specific project for a job is done with --project = ProjectName switch

If no project name is given the default project for the user is used

- In case of several projects a reverse sort is applied to the list of projects and the top project is selected as the default one
  - 14b015, 14a005, 14b025, 14b005 → 14b025 is the default project

On a computer node the name of the project is stored in OAR_PROJECT_NAME variable

OAR queues

Prioritization of the jobs and used scheduler is highly depend of the nature of your job. Jobs with high walltime have lower priority then short jobs

admin

- priority = 10
- scheduler = timesharing_and_fairsharing

development

- priority = 9
- scheduler =timesharing

short

- priority = 7
- scheduler = timesharing_and_fairsharing

medium

- priority = 5
- scheduler = timesharing_and_fairsharing

long

- priority = 3
- scheduler = timesharing_and_fairsharing

default

- priority = 2
- scheduler = timesharing_and_fairsharing

besteffort

- priority = 0
- scheduler = timesharing_and_fairsharing

OAR scheduler

timesharing - This is the default OAR scheduler. Simple FIFO algorithm

timesharing_and_fairsharing – Same as the default OAR scheduler but with an important addition

This algorithm looks at the consumption past of the user and the project and try to order waiting jobs with fairsharing in mind

Fairsharing is based on a parameter called Karma

In a nutshell : high Karma means high waiting time for your jobs

OAR queues

You can specify the queue name with: -q queue_name switch

- Automatic queue routing can override the value specified by a user

Automatic queue routing is taking into account the walltime value specified in job description

- **development** : 2 hours
- **short** : 12 hours
- **medium** : 48 hours
- **long** : 168 hours (a week) not available for SMP jobs

If you need the **development** or the **besteffort** queue you must specify the name of the queue explicitly

Development and besteffort queues

Jobs in the **besteffort** queue can be killed at any moment. Therefor these jobs can use any available resource in a given moment of time

- Ideal for massive Monte-Carlo simulations or any other kind of jobs which can be suddenly interrupted

Certain resources can be attached to a specific queue

Resources with property shortnode=YES are reserved for the jobs in the **development** queue

Properties can be assigned, changed or removed automatically

- During the working hours 40 nodes are reserved for development
- During the weekends only 10 nodes reserved for development queue
- Reservation is removed at midnight so all nodes are accessible for long term jobs

Job submission

The user can submit a job with command oarsub

- Passive jobs – OAR sends a script on execution on requested resources
- Interactive jobs – OAR is returning a login shell on requested resources
  - Ideal for debugging purpose

```
oarsub -p "smp and nodetype='SMP512Gb'" -l host=3,walltime=47:59:00 --project
11a011 script_name
```

- passive job
- 3 hosts for 47 hours – long queue
- Project to account used resources is 11a011
- Requested resources is a smp machine with property SMP512Gb

```
oarsub -l nodes=1/core=4,walltime=1:59:00 -p "host='node088'" -q development -I
```

- Interactive job
- 4 cores on a single node for 2 hours in the development queue
- Requested resource is a specific machine : node088

Job submission

To connect to already running job use -C switch

```
oarsub -C 323847
```

- Interactive job

Request that the job starts at a specified time

```
oarsub -r "2014-12-01 11:00:00" -l /nodes=12/core=6
script_name
```

Job reservation status

- **none**: the job has no reservation
- **toSchedule**: the job has a reservation and must be approved by the scheduler
- **scheduled**: the job has a reservation and it's scheduled by OAR

Parametric job submission

Submit an array job with 10 identical subjobs:

```
oarsub -l /nodes=4 /home/users/toto/prog --array 10
```

Parametric job with parameters stored in a file params.txt

```
# my param file
#a subjob with single parameter
100
#a subjob without parameters
""
#a subjob with string containing spaces as delimiter for parameters
"arg1a arg1b arg1c" "arg2a arg2b"
```

OAR generates 3 jobs and a special identifier called OAR_ARRAY_ID

```
oarsub /home/test/prog --array-param-file /home/test/params.txt
```

- OAR_JOB_ID=323848
- OAR_JOB_ID=323849
- OAR_JOB_ID=323850
- OAR_ARRAY_ID=323848

Job submission

User can prepare a script with OAR directives which can be scanned during script submission. The script has to have exec permissions

```
chmod +x /home/username/script.oar
```

Script example (file /home/username/script.oar):

```
#!/bin/bash
#OAR -n test
#OAR --notify mail:matvey.sapunov@univ-amu.fr
#OAR -l nodes=2/core=8,walltime=50:00:00
#OAR -p cluster
#OAR --project 14a026
#OAR -O OAR.%jobid%.out
#OAR -E OAR.%jobid%.err
/home/username/program
```

Submit the script :

```
oarsub -S /home/username/script.oar
```

Job notifications

User notification can be done via e-mail or a script

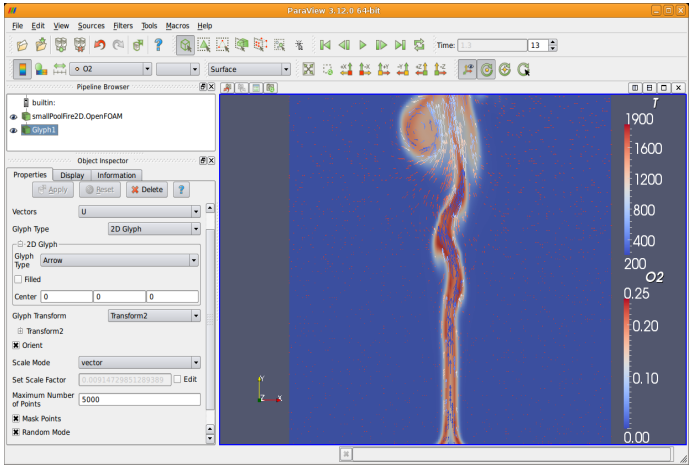
- The user wants to receive an email
  - The syntax is "**mail**:name@domain.com".
  - The subject of the mail is of the form: *OAR* [TAG]: job_id (job_name) on OAR_server_hostname
- The user wants to launch a script:
  - The syntax is "**exec**:/path/to/script args".
  - OAR server will connect (using OPENSSH_CMD) on the node where the oarsub command was invoked and then launches the script with the following arguments : job_id, job_name, TAG, comments

TAG can be:

- **RUNNING** : when the job is launched
- **END** : when the job is finished normally
- **ERROR** : when the job is finished abnormally
- **INFO** : used when oardel is called on the job
- **SUSPENDED** : when the job is suspended
- **RESUMING** : when the job is resumed

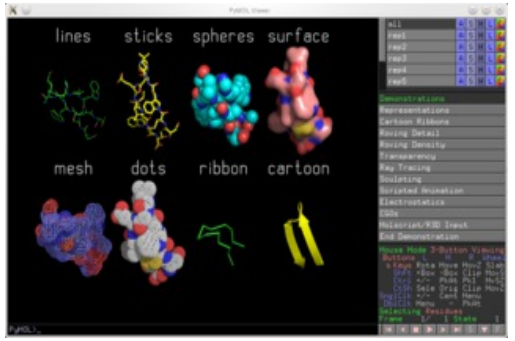
Visualisation job

Special type of job dedicated for visualisation. Can execute a 3-D application with GUI like OpenFOAM, Molekel etc



From the front-end, to ask for a visualisation session:

```
[user@login ~]$ visu_sub.sh
[ADMISSION RULE] Modify resource description with type constraints
OAR_JOB_ID=559
Waiting job 559 to be running.
You can launch your VNC viewer on the address:
visu.ccamu.u-3mrs.fr:11
Password: 28405608
Note: This password is only valid ONE time. If you want to generate another password for this session then type:
OAR_JOB_ID=559 oarsh visu vncpasswd -o -display visu:11
[user@login ~]$
```



Visualisation job

To connect, you need a VNC client. We advise you to use tigervnc version 1.2 or higher

From your local machine, start tigervnc and connect to the indicated address given at the submission time and with the associated password

Hostname: **visu.ccamu.u-3mrs.fr:11**

Password: **28405608**

It is possible to connect several people simultaneously on the same session (each connection needs a different password).

To ask for a new password (from the front-end):

```
OAR_JOB_ID=559 oarsh visu vncpasswd -o -display visu:11
```

By default, tigervnc does not accept the sharing, it is important to tick the option **Shared** (don’t disconnect other viewers)

On the visualisation node, to start a 3D application from the shell terminal:

```
[user@login ~]$ vglrun /chemin/vers/mon/application
```

Admission rules

OAR offers a powerful system letting system administrators customize the way that jobs enter into queues (or are rejected from queues)

An admission rule is a little perl script executed on a frontend before job send to OAR server

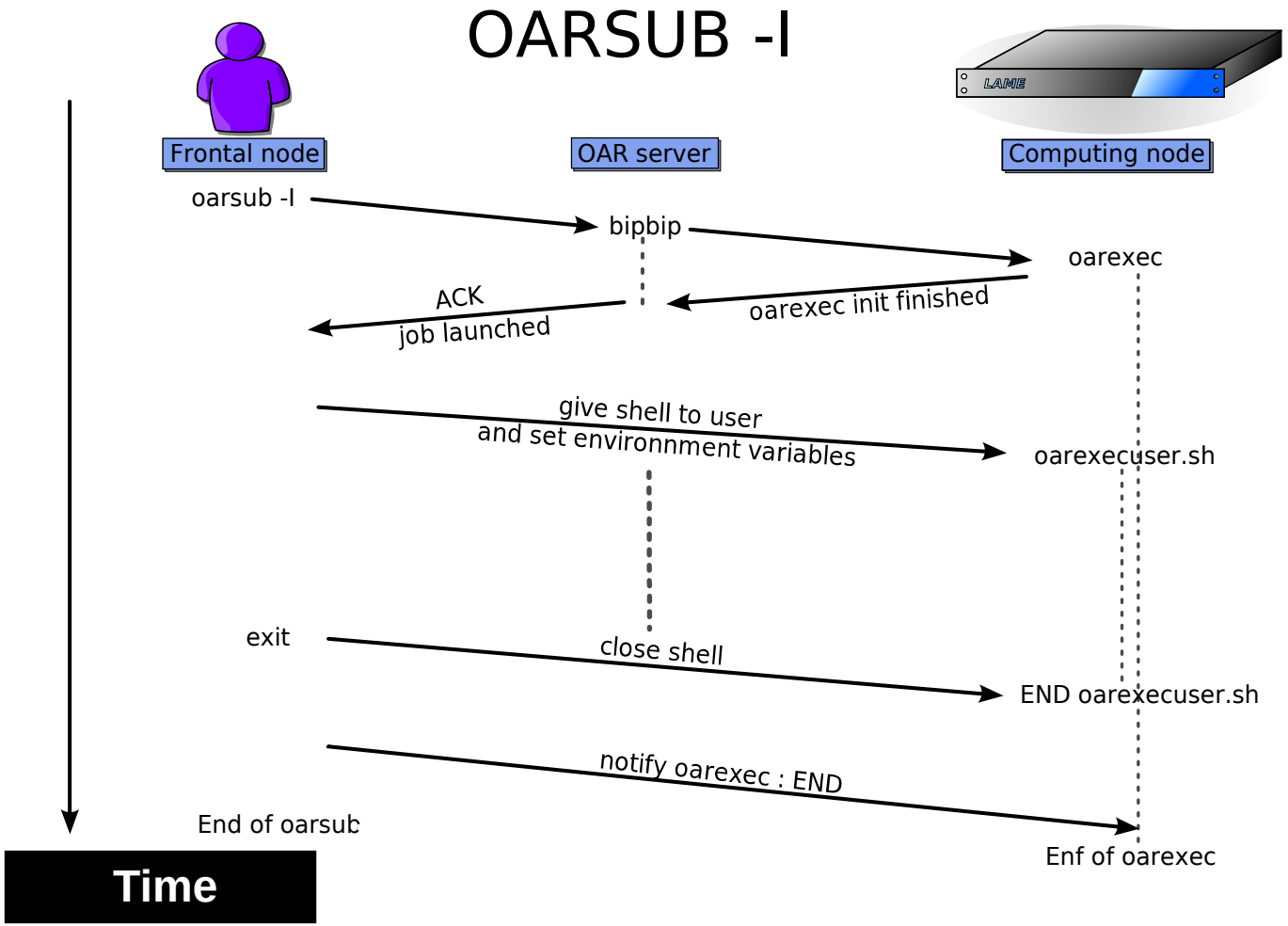
Data for the rules are provided by oarsub command

```
Rule : 12

# Title: Default walltime

# Description: Give a default walltime if none is given

my $default_wall = OAR::IO::sql_to_duration("0:30:00");

foreach my $mold (@{$ref_resource_list}){

    if (!defined($mold->[1])){

        print("[ADMISSION RULE] Set default walltime to $default_wall.\n");

        $mold->[1] = $default_wall;

    }

}
```

Job submission

```
> oarsub -l nodes=1/core=4,walltime=1:59:00 -p "host='node088'" -q development -I
[ADMISSION RULE] Modify resource description with type constraints
   [INTERACTIVE NODES] This job is granted into the nodes for short or interactive routes
[JOB PROJECT] No project selected using rheticus
[JOB START] Your job was successfully submited with the job number ($OAR_JOB_ID): 323848
Interactive mode : waiting...
Starting...


Initialize X11 forwarding...
Connect to OAR job 323848 via the node node088
```

Job execution



OAR job states

- **Waiting**: the job is waiting OAR scheduler decision
- **Hold**: user or administrator wants to hold the job. So it will not be scheduled by the system
- **toLaunch**: the OAR scheduler has attributed some nodes to the job. So it will be launched
- **toError**: something wrong occurred and the job is going into the error state
- **toAckReservation**: the OAR scheduler must say "YES" or "NO" to the waiting oarsub command because it requested a reservation
- **Launching**: OAR has launched the job and will execute the user command on the first node
- **Running**: the user command is executing on the first node
- **Suspended**: the job was in Running state and there was a request to suspend this job. In this state other jobs can be scheduled on the same resources
- **Finishing**: the user command has terminated and OAR is doing work internally
- **Terminated**: the job has terminated normally
- **Error**: a problem has occurred

Job monitoring

To show information about a job or set of jobs use oarstat command

Status of the job

```
oarstat -sj 323847

323847: Terminated
```

Job's event

```
oarstat -ej 323847

2014-11-30 19:09:32| 323847| SWITCH_INTO_TERMINATE_STATE: [bipbip
323847] Ask to change the job state
```

Information about the job

```
oarstat -j 323847

Job id     Name           User           Submission Date     S Queue

323847     interactive    msapunov       2014-11-30 19:07:49 T develop-
men
```

Job details : oarstat -fj 323847

```
Job_Id: 323847
    job_array_id = 323847
    job_array_index = 1
    name = interactive
    project = rheticus
    owner = msapunov
    state = Terminated
    wanted_resources = -l "{type =
'default'}/host=1/core=4,walltime=1:59:0"
    types =
    dependencies =
    assigned_resources = 1045+1046+1047+1048
    assigned_hostnames = node088
    queue = development
    command =
    launchingDirectory = /home/msapunov
    stdout_file = OAR.interactive.323847.stdout
    stderr_file = OAR.interactive.323847.stderr
    jobType = INTERACTIVE
    properties = ((host='node088') AND
cluster='YES') AND host NOT IN ('gpu001')
    reservation = None
    walltime = 1:59:0
    submissionTime = 2014-11-30 19:07:49
    startTime = 2014-11-30 19:07:50
    stopTime = 2014-11-30 19:09:32
    cpuset_name = msapunov_323847
    initial_request = oarsub -l
nodes=1/core=4,walltime=1:59:00 -p host='node088'
-q development -I
    message = FIFO scheduling OK
    scheduledStart = no prediction
    resubmit_job_id = 0
    events = [2014-11-30 19:09:32]
SWITCH_INTO_TERMINATE_STATE:[bipbip 323847] Ask to
change the job state
```

Accounting

Accounting information between two dates

```
oarstat --accounting '2014-11-18, 2014-11-19' -u msapunov

Usage summary for user 'msapunov' from 2014-11-18 to 2014-11-19:

Start of the first window:   2014-11-17 01:00:00

End of the last window:      2014-11-19 00:59:59

Asked consumption:           897800 ( 10 days 9 hours 23 minutes 20 seconds )

Used consumption:            259704 ( 3 days 8 minutes 24 seconds )

By project consumption:

  rheticus:

    Asked :                  897800 ( 10 days 9 hours 23 minutes 20 seconds )

    Used :                   259704 ( 3 days 8 minutes 24 seconds )

    Last Karma :             Karma = 0.003
```

Important note : consumption = walltime * number of cores

Useful commands

The command to delete or to checkpoint the job(s) : **oardel**

- `oardel 323848 323849`
  - Delete two jobs 323848 and 323849
- `oardel -c 323849`
  - Send a checkpoint signal to the job 323849 (type of signal defined as oarsub option)

User can hold a job in OAR batch scheduler with command **oarhold**

- Remove a job from the scheduling queue if it is in the "**Waiting**" state
- Suspend a job if it is in "**Running**" state, sending SIGINT signal

Ask OAR to change a job states into "**Waiting**" when it is on "**Hold**" or in "**Running**" if it is "**Suspended**" state with **oarresume** command

Où trouver plus d'informations ?

- **Site web du mésocentre : equipex-mesocentre.univ-amu.fr**
  - Informations de présentation
  - Liste des logiciels, Tutoriaux
  - Accès à Monika
  - À venir une section Suivi d'activité

- **Liste de diffusion : equipex-mesocentre@univ-amu.fr**

- **Comité technique :**
  - equipex-mesocentre-techn@univ-amu.fr
  - +33 (0)4 13 55 12 15 / 55 03 33

Questions / réponses :